INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Sequencing of parts in a robotic cell

Haoxun CHEN - Chengbin CHU - Jean-Marie PROTH

N° 2496
Février 1995

PROGRAMME 5

SEQUENCING OF PARTS IN A ROBOTIC CELL

Ordonnancement de pièces dans une cellule robotisée

Chen Haoxun [1] [2] **Chu Chengbin** [1] **Proth Jean-Marie** [1],[3]

1. INRIA-Lorraine, CESCOM, Technopole de Metz 2000, 57070 Metz, France
2. Systems Engineering Institute, Xi'an Jiaotong University, Xi'an 710049, P.R. China
3. Institute for Systems Research, University of Maryland, U.S.A.

Abstract. This paper considers scheduling problems in a robotic cell which produces a set of parts on several machines served by a robot. We study the problem of sequencing parts in the cell in order to minimize the production cycle time when the sequence of the robot moves is given. This problem is NP-hard for most of the one-unit robot move cycles in a robotic cell with more than two machines producing more than two part-types. We first give a mathematical formulation to the problem, and then propose a branch-and-bound algorithm to solve it. The bounding scheme of this algorithm is based on relaxing, for all the machines except two, the constraint that the machine is occupied by a part for a period at least as long as the processing time of the part. It turns out that the lower bound obtained in this way is tight. This relaxation allows us to overcome the complexity of the problem. The lower bound can be computed using the algorithm of Gilmory and Gilmore. Computational experiments on part sequencing problems in three-machine robotic cells are given.

Keywors. Scheduling, Robotic Cell, Lower Bounds, Branch and Bound.

Résumé : Ce rapport considère les problèmes d'ordonnancement dans une cellule dans laquelle sont produites des pièces sur des machines servies par un robot. Nous étudions le problème qui consiste à minimiser le temps de cycle pour une séquence donnée de mouvements du robot. Ce problème a été prouvé NP-difficile pour la plupart des séquences 1-périodiques de mouvements du robot dans une cellule comportant plus de deux machines sur lesquelles sont produits plus de deux types de pièces. Nous fournissons d'abord une formulation mathématique du problème et proposons ensuite une procédure par séparation et évaluation pour sa résolution. Le calcul des bornes inférieures est basé sur la relaxation qui consiste à annuler les durées opératoires des pièces sur toutes les machines sauf deux. Il s'avère que les bornes inférieures ainsi obtenues sont de bonne qualité. Cette relaxation nous permet de surmonter le problème de la complexité, puisque les bornes inférieures peuvent être obtenues en appliquant l'algorithme de Gilmore et Gomory en temps polynômial. Des expériences numériques sur l'ordonnancement de pièces dans une cellule à trois machines sont également fournies dans ce rapport.

Mots clefs : Ordonnancement, Cellule robotisée, Bornes inférieures, Séparation et évaluation.

1. Introduction

Industrial robots play an important role in advanced manufacturing systems. A major application of industrial robots is in so-called robotic cells, which consist of one or more machines, a robot which loads and unloads parts on machines, and a material handling mechanism for feeding the cell and removing parts from the cell. Optimizing the performance of such cells is a major issue. Since the robot in such a cell usually performs repetitive sequences of pickup, move, load, unload, and drop operations, the performance of the cell depends on the scheduling of robot activities and the operations to be performed on the parts.

Only a few studies on the scheduling of parts and robot moves in robotic cells have been reported in the literature. An extensive review of the literature related to this problem is given by Sethi et al. [8] and further references can be found in the paper of Hall, Kamoun and Sriskandarajah [3]. Sethi et al. [8] study the problem of determining the robot move cycle and the sequence of parts in a minimal part set (MPS) that jointly minimize the production cycle time. They find the optimal sequence of robot moves for a two machine robot-centered cell producing a single part-type. For the case of three machines and one part type, they show that under specific assumptions about the data, two of the six potentially optimal robot move cycles are dominated by the other four. Finally, for the case of two machines and multiple part-types, they find a polynomial time algorithm to solve the problem of sequencing parts for a given sequence of robot moves. Hall, Kamoun and Sriskandarajah [3] consider the same problem. They provide an efficient algorithm that simultaneously optimizes the robot move and part sequencing problems. For single part-type problems, they prove that in a very general environment, the repetition of one-unit cycles dominates more complicated policies that produce two or more units. Hall, Kamoun and Sriskandarajah [4] consider the problem with three or more machines. They prove that, in a robotic cell with m machines, the part sequencing problem can be polynomially solved for $1+m(m-1)/2$ out of $m!$ one-unit robot move cycles, and is unary NP-hard for the other cycles. They propose a pseudo-polynomial algorithm for the part sequencing problem when the number of possible part-types is fixed. Kamoun, Hall and Sriskandarajah [5] propose several heuristics for robotic cells with three machines.

Although optimal one-unit robot move cycle is in general not an optimal solution, if we consider more complicated policies that produce two or more units (see Hall, Kamoun and Sriskandarajah [3]), it is easily computed and applied due to its simply repetitive nature. For this reason, we confine ourselves to one-unit robot move cycles in this paper. Since the number of machines in a robotic cell is usually no larger than four, the number of possible one unit robot move cycles is small enough to be able to find an optimal one by enumeration. Thus, the remaining problem is to determine an optimal part sequence when the sequence of the robot moves is specified.

To solve this problem, it appears that an implicit enumeration of feasible solutions is unavoidable. In this paper, we first give a mathematical formulation to this problem, and then propose a branch and bound algorithm to solve it. The bounding scheme of this algorithm is obtained by relaxing the constraint that each machine must process a part for at least its processing time on the machine for all machines but two. We show that if the two machines are chosen as two successive ones, the corresponding relaxed problem can be solved by using the well known algorithm of Gilmore and Gomory for a special solvable class of the traveling saleman problems in O(nlogn) time. The relaxed problems for some of the other choices, however, are likely to be NP-hard. For this reason, each pair of the two machines chosen for computing the lower bound is composed of two successive machines. Computational experiments on intractable part sequencing problems in three-machine robotic cells are given.

2. Mathematical Formulation

We first introduce some notations required to specify a general robotic cell considered in this paper.

m	number of machines in the robotic cell. We denote these machines by M_i, i=1, 2, ..., m, input station by I (or M_0) and output station by O (or M_{m+1}). Parts are processed succesively by M_1, M_2,..., M_m.
k	number of part types produced in the cell. These part types are denoted respectively by 1, 2, ..., k.
$r_1, r_2, \ldots, r_k$	smallest integers such that $r_i / \sum_{j=1}^{k} r_j$ is the production ratios required for the part type i.
MPS	minimal part set consisting of r_i parts of part type i, i = 1, 2, ..., k.
n	total number of parts to be produced in the MPS, i.e., $n=r_1+r_2+\ldots+r_k$.
$p_{i,j}$	processing time of part i on machine M_j, i=1, 2, ..., n, j=1, 2, ..., m.
$\theta_{i,j}$	time needed for the robot to travel from M_i to M_j, $i \neq j$, i, j =0, 1, ..., m, m+1.
δ_i^-	time needed for the robot to pick up a part from M_i, i=0, ..., m.
δ_i^+	time needed for the robot to drop a part onto M_i, i=1, ..., m+1.
α_i	activity of the robot consisting of picking up a part from M_i, transporting the part to M_{i+1} and droping the part onto M_{i+1}, i=0,...,m.
$\varepsilon_{i,j}$	activity of the robot consisting of traveling from M_i to M_j without carrying a part (empty move).
$\bar{\theta}_i$	time needed by robot activity α_i, i.e., $\bar{\theta}_i = \delta_i^- + \theta_{i,i+1} + \delta_{i+1}^+$.

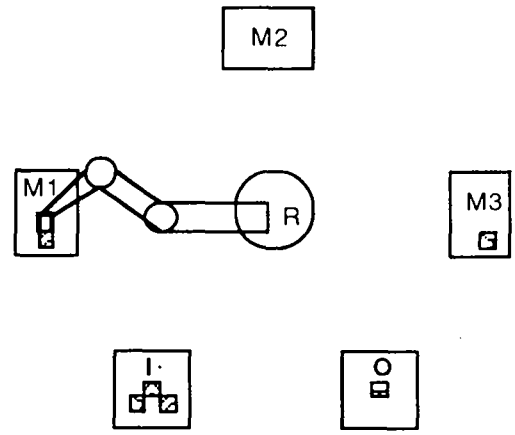


Fig. 1. A robotic cell with three machines

Figure 1 illustrates a robotic cell with three machines.

A one-unit robot move cycle can be represented as a sequence of the form:

$$\sigma_R = \alpha_{i_0} \varepsilon_{i_0+1,i_1} \alpha_{i_1} \varepsilon_{i_1+1,i_2} \alpha_{i_2} \ldots . \varepsilon_{i_{m-1}+1,i_m} \alpha_{i_m} \varepsilon_{i_m+1,i_0}$$

or briefly

$$\sigma_R = \alpha_{i_0} \alpha_{i_1} \alpha_{i_2} \ldots . \alpha_{i_m}$$

where $i_0, i_1, \ldots, i_m$ is a permutation of 0, 1, ... , m.

Because of the repetitive nature of the one-unit robot move cycles, we can assume that $i_0=0$ without loss of generality.

Figure 2 illustrates a one-unit robot move cycle for the robotic cell represented in Fig. 1. This cycle can be represented as $\alpha_0\alpha_2\alpha_1\alpha_3$.

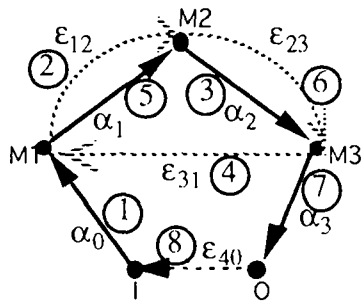


Fig. 2. A one-unit robot move cycle

When we consider the sequence of robot moves as well as the sequence of parts in a robotic cell, a time diagram is very helpful to understand all activities of the machines and the

robot in one production cycle of the robotic cell. To illustrate such a time diagram, let us consider the robotic cell as shown in Fig. 1. Suppose that the cell produces three parts in one cycle, one for each part type (the three parts are numbered by 1, 2, 3 respectively), that the part sequence is 1, 2, 3 and that the robot move sequence is specified as in Fig. 2. Then, the time diagram of one production cycle in the robot cell is illustrated in Fig. 3, where a box labelled i/j represents the activity that the robot unloads a part i from machine M_j, transports the part to machine M_{j+1} and then loads the part onto machine M_{j+1}. Robot moves related to the parts that are delivered into the cell during this robot cycle (resp. last robot cycle) are represented by boxes on layer 0 (resp. layer -1).

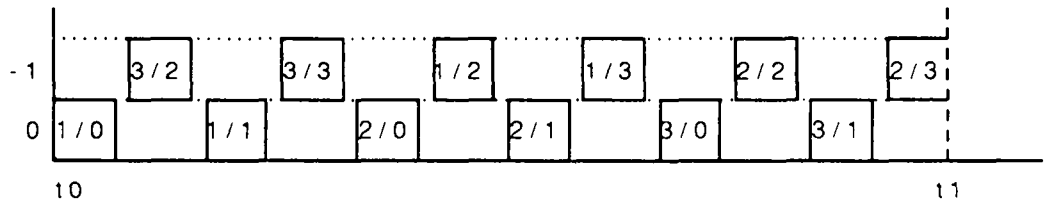


Fig. 3. Diagram of the robot activities during a production cycle.

For a robotic cell with three machines, there are six possible one-unit cycles of the robot moves S_1, S_2, ..., S_6 (see Sethi, et. al. [8]) which can be described as:

$S_1 = \alpha_0\alpha_1\alpha_2\alpha_3$;
$S_2 = \alpha_0\alpha_2\alpha_1\alpha_3$;
$S_3 = \alpha_0\alpha_1\alpha_3\alpha_2$;
$S_4 = \alpha_0\alpha_3\alpha_1\alpha_2$;
$S_5 = \alpha_0\alpha_2\alpha_3\alpha_1$;
$S_6 = \alpha_0\alpha_3\alpha_2\alpha_1$.

It has been proved by Hall, et. al. [4] that although the part sequencing problems for four robot sequences S_1, S_3, S_4 and S_5 are polynomial time solvable, the part sequencing problems for the other two robot sequences S_2 and S_6 are unary NP-complete.

Without loss of generality, we assume that there are n different parts to be scheduled at input station I. In other words, each of the r_i parts of type i in the MPS is considered as a specific part. Suppose that the n parts are numbered by 1,2,...,n. Then, a cyclic schedule of parts can be represented as a permutation $\sigma = (\sigma(1), ..., \sigma(n))$ of parts 1, 2, ..., n, where $\sigma(i)$ means that part $\sigma(i)$ is scheduled at i-th position. Let T_σ denote the cycle time related to σ. Our objective is to find a permutation σ which minimizes the cycle time T_σ.

Given a complete cyclic schedule $\sigma = (\sigma(1), \ldots, \sigma(n))$, the cycle time T_σ can be expressed as:

$$T_\sigma = T_{\sigma(1)\sigma(2)} + T_{\sigma(2)\sigma(3)} + \ldots + T_{\sigma(n)\sigma(1)}, \quad (1)$$

where $T_{\sigma(i)\sigma(i+1)}$ is the time required to produce part $\sigma(i)$ which is measured from the time that the robot starts to pick up part $\sigma(i)$ at input station to the time that the robot starts to pick up part $\sigma(i+1)$ at input station.

Let us denote by $w_{i,j}$ the robot waiting time in front of machine M_j to pick up part $\sigma(i)$, i=1, 2, ..., n, j=1, 2, ..., m. Conventionally, $\sigma(i)$ is sometimes denoted by [i] and we assume that $\sigma(i)$, $w_{i,j}$, $j \in \{1, \ldots, m\}$ are defined for any integer i, and that $\sigma(i) = \sigma(k)$, $w_{i,j} = w_{k,j}$ if (i-k) is a multiple of n.

In the following, we establish a mathematical formulation for part sequencing problems in a 3-machine robotic cell when robot sequence S_2 or S_6 are given.

The notion of production cycle will be used: It is the time period needed to produce the parts belonging to an MPS. Note that the production cycle is different from the one-unit robot move cycle.

(a) Part sequencing in a 3-machine robotic cell with robot sequence S_2

The time diagram of one production cycle in a 3-machine robotic cell with robot sequence S_2 is shown in Fig. 4, where [i] represents the part scheduled at the i-th position, a box labelled as [i]/j represents the robot activity which consists of picking up part [i] from machine M_j, transporting the part to machine M_{j+1} and then dropping the part onto machine M_{j+1}. From now on, such an activity is referred to as robot move [i]/j.

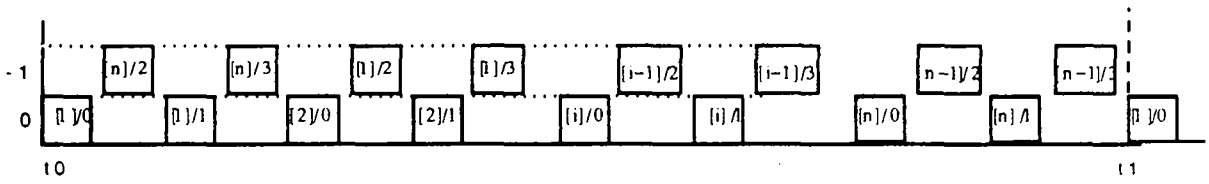


Fig. 4. Time diagram of a cycle in a 3-machine robotic cell with robot sequence S_2

Let $C = \bar{\theta}_0+\theta_{12}+\bar{\theta}_2+\theta_{31}+\bar{\theta}_1+\theta_{23}+\bar{\theta}_3+\theta_{40}$, $\mu_1 = \theta_{12}+\bar{\theta}_2+\theta_{31}$, $\mu_2 = \theta_{23}+\bar{\theta}_3+\theta_{40}+\bar{\theta}_0+\theta_{12}$, $\mu_3 = \theta_{31}+\bar{\theta}_1+\theta_{23}$. C is the total robot busy time in one robot move cycle. μ_i is the total robot busy time between the time that a part is loaded onto machine i and the time that the part is unloaded from machine i, i=1, 2, 3.

We can derive a formula for $T_{\sigma(i)\sigma(i+1)}$ as follows:

$$T_{\sigma(1)\sigma(2)} = C+w_{n,2}+w_{1,1}+w_{n,3}, \tag{2a}$$
$$T_{\sigma(i)\sigma(i+1)} = C+w_{i-1,2}+w_{i,1}+w_{i-1,3}, \quad i=2, \ldots, n. \tag{2b}$$

Thus,

$$T_\sigma = \sum_{i=1}^{n} T_{\sigma(i)\sigma(i+1)} = nC + \sum_{i=1}^{n} (w_{i,1}+w_{i,2}+w_{i,3}) \tag{3}$$

subject to

$$w_{n,2} \geq \max\{0, p_{[n],2} - \mu_2 - w_{n-1,3}) \tag{4a}$$
$$w_{1,1} \geq \max\{0, p_{[1],1} - \mu_1 - w_{n,2}) \tag{4b}$$
$$w_{n,3} \geq \max\{0, p_{[n],3} - \mu_3 - w_{1,1}) \tag{4c}$$
$$w_{i-1,2} \geq \max\{0, p_{[i-1],2} - \mu_2 - w_{i-2,3}), \quad i=2, \ldots, n, \tag{4d}$$
$$w_{i,1} \geq \max\{0, p_{[i],1} - \mu_1 - w_{i-1,2}), \quad i=2, \ldots, n, \tag{4e}$$
$$w_{i-1,3} \geq \max\{0, p_{[i-1],3} - \mu_3 - w_{i,1}), \quad i=2, \ldots, n, \tag{4f}$$

or equivalently,

$$w_{n,2} + w_{n-1,3} \geq p_{[n],2} - \mu_2, \tag{5a}$$
$$w_{1,1} + w_{n,2} \geq p_{[1],1} - \mu_1, \tag{5b}$$
$$w_{n,3} + w_{1,1} \geq p_{[n],3} - \mu_3, \tag{5c}$$
$$w_{i-1,2} + w_{i-2,3} \geq p_{[i-1],2} - \mu_2, \quad i=2, \ldots, n, \tag{5d}$$
$$w_{i,1} + w_{i-1,2} \geq p_{[i],1} - \mu_1, \quad i=2, \ldots, n, \tag{5e}$$
$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \quad i=2, \ldots, n, \tag{5f}$$
$$w_{i,1}, w_{i,2}, w_{i,3} \geq 0, \quad i=1, \ldots, n. \tag{5g}$$

(4a) means that the robot cannot pick up part [n] from machine 2 until the part finishes its processing on the machine. The meanings of (4b) — (4f) are similar.

Therefore, the part sequencing problem for S_2 can be formulated as follows:

P_2:

$$\min_{\sigma,w} T_\sigma = nC + \sum_{i=1}^{n} (w_{i,1}+w_{i,2}+w_{i,3})$$

subject to (5a) — (5g), where $w=\{w_{i,j}, i=1, 2, \ldots, n, j=1, 2, 3\}$.

(b) Part sequencing in a 3-machine robotic cell with robot sequence S_6

The time diagram of one production cycle in a 3-machine robotic cell with robot sequence S_6 is shown in Fig. 5.

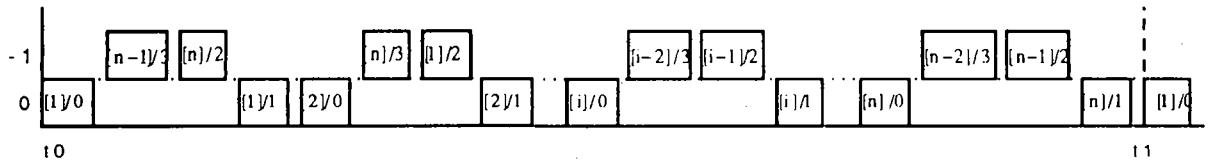


Fig. 5. Time diagram of a cycle in a 3-machine robotic cell with robot sequence S_6

Let $C = \bar{\theta}_0+\theta_{13}+\bar{\theta}_3+\theta_{42}+\bar{\theta}_2+\theta_{31}+\bar{\theta}_1+\theta_{20}$, $\mu_1 = \theta_{13}+\bar{\theta}_3+\theta_{42}+\bar{\theta}_2+\theta_{31}$, $\mu_2 = \theta_{20}+\bar{\theta}_0+\theta_{13}+\bar{\theta}_3+\theta_{42}$, $\mu_3 = \theta_{31}+\bar{\theta}_1+\theta_{20}+\bar{\theta}_0+\theta_{13}$. The meanings of C and μ_i, i=1,2,3 are the same as these in (a).

Similar to (a), the part sequencing problem for S_6 can be formulated as follows:

P_6:

$$\min_{\sigma,w} T_\sigma = nC + \sum_{i=1}^{n}(w_{i,1}+w_{i,2}+w_{i,3})$$

subject to

$$w_{n-1,3} + w_{n,1} \geq p_{[n-1],3} - \mu_3, \tag{6a}$$

$$w_{n,2} + w_{n-1,3} \geq p_{[n],2} - \mu_2, \tag{6b}$$

$$w_{1,1} + w_{n,2} + w_{n-1,3} \geq p_{[1],1} - \mu_1, \tag{6c}$$

$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \quad i=1, \ldots, n-1, \tag{6d}$$

$$w_{i,2} + w_{i-1,3} \geq p_{[i],2} - \mu_2, \quad i=1, \ldots, n-1, \tag{6e}$$

$$w_{i+1,1} + w_{i,2} + w_{i-1,3} \geq p_{[i+1],1} - \mu_1, \quad i=1, \ldots, n-1, \tag{6f}$$

$$w_{i,1}, w_{i,2}, w_{i,3} \geq 0, \quad i=1, \ldots, n. \tag{6g}$$

3. Lower bounds

The enumeration scheme used in almost all branch-and-bound algorithms developed so far generates all n! permutation schedules in the following way: A node at the l-th level of the search tree is characterized by a *partial schedule* $\sigma = (\sigma(i), i \in S)$, where S is a subset of $\{1, 2, \ldots, n\}$, $|S| = l$, $\sigma(i)$ indicates that part $\sigma(i)$ is scheduled at the i-th position, for $i \in S$. Any permutation $\bar{\sigma}$ of the index set $\bar{S}$ ($\bar{S} = \{1, 2, \ldots, n\} \backslash S$) of the unscheduled jobs is a complement of σ, i.e., a complete permutation schedule $\sigma \circ \bar{\sigma} = (\sigma(i), i \in S; \bar{\sigma}(i), i \in \bar{S})$, where

"$\circ$" denotes the concatenation, as usual.

In the branch-and-bound algorithm presented hereafter, a partial schedule is represented by $\sigma=(\sigma(i), i\in S)$, where $S = \{n\}$ at the 0-th level of the search tree (i.e., at the root node), $S=\{n, 1\}$ at the 1-th level, $S = \{n, 1, 2, \ldots, k\}$, $2\leq k\leq n-1$, at the k-th level. Note that due to the cyclic nature of the schedules we consider, we can arbitrarily specify $\sigma(n)$, the part scheduled at the last position, at the beginning of a run of the algorithm. This is the reason why we take $S=\{n\}$ at the 0-th level, $S = \{n, 1\}$ at the 1-th level, ... Conventionally, we set $k=0$ if $S = \{n\}$ and $k=1$ if $S = \{n, 1\}$.

Given a partial schedule σ, we should derive a lower bound of the cycle time of $\sigma\circ\overline{\sigma}$ for all possible σ. One approach is to obtain such a lower bound by relaxing the processing time requirements of parts on each machine using Lagrangian multipliers and then to calculate the lower bound by using subgradient method. We can expect that the Lagrangian-relaxation-based lower bound is sharp, but it is at the expense of excessive computation time. For this reason, we focus on finding a lower bound making a trade-off between the sharpness and the computational requirements.

The basic idea is to obtain a lower bound by relaxing the processing time requirements of the unscheduled parts on some machines, i.e., by assuming that, for each of the machines, when the robot has been in front of the machine and ready to unload the part being processed, the robot can unload the part even if the previous part has not finished its processing on the machine. The remaining machines on which the processing time requirements of parts are not relaxed are referred to as *bottleneck* machines.

As mentioned in the introduction, any problem involving three or more bottleneck machines for NP-hard part sequencing problems in robotic cells is likely to be NP-hard. For this reason, we restrict ourselves to choosing two machines as bottleneck machines.

Let $LB_\sigma(u,v)$ denote the lower bound found by choosing machines M_u and M_v, $1\leq u<v\leq m$, as the two bottleneck machines (i.e., by relaxing the processing time requirements of the unscheduled parts on machines belonging to $\{1, 2, \ldots, m\}\backslash\{u, v\}$). Then, $LB_\sigma(\Omega) = \max\{LB_\sigma(u, v) \mid (u, v)\in\Omega\}$ is a valid lower bound for any $\Omega\subseteq Z$, where $Z = \{(u, v) \mid 1\leq u<v\leq m\}$

In the following, we first introduce the lower bound scheme for robot sequences S_2 and S_6 in a 3-machine robotic cell and then extend it to general robotic cells with an arbitrarily robot sequence.

3.1. 3-machine robotic cell with robot sequence S_2

(a) $LB_\sigma(1,2)$

In this case, M_1 and M_2 are taken as the two bottleneck machines. Note that T_σ can be rewritten as

$$T_\sigma = nC + \sum_{i=1}^{n} (w_{i-2,3}+w_{i-1,2}+w_{i,1}) \tag{7}$$

and the constraints (5a) — (5g) can be rewritten as

$$w_{i-2,3} + w_{i-1,1} \geq p_{[i-2],3} - \mu_3, \qquad i=1, 2, \ldots, n, \tag{8a}$$
$$w_{i-1,2} + w_{i-2,3} \geq p_{[i-1],2} - \mu_2, \qquad i=1, 2, \ldots, n, \tag{8b}$$
$$w_{i,1} + w_{i-1,2} \geq p_{[i],1} - \mu_1, \qquad i=1, 2, \ldots, n, \tag{8c}$$
$$w_{i-2,3}, w_{i-1,2}, w_{i,1} \geq 0, \qquad i=1, 2, \ldots, n. \tag{8d}$$

If we drop constraints (8a) for i=1 and i=k+1, ..., n, i.e., relax the processing time requirements of parts $\sigma(i)$, i=k-1, k, ..., n-1 on machine 3, we can obtain a relaxed problem $\hat{P}_2$ of P_2 (P_2 is defined in Section 2 (a)) as follows:

$\hat{P}_2$: $\min nC + \sum_{i=1}^{n} (w_{i-2,3}+w_{i-1,2}+w_{i,1})$

subject to (8a), i=2, ..., k; (8b)—(8d), i=1, ..., n.

$\hat{P}_2$ can be further decomposed into the following two subproblems P_σ^s and P_σ^u:

P_σ^s: $\min kC + \sum_{i=1}^{k} (w_{i-2,3}+w_{i-1,2}+w_{i,1})$

subject to (8a), i=2, ..., k; (8b)—(8d), i=1, ..., k,

and

P_σ^u: $\min (n-k)C + \sum_{i=k+1}^{n} (w_{i-2,3}+w_{i-1,2}+w_{i,1})$

subject to (8b) — (8d), i=k+1, ..., n.

The sum of the minimal objective values of P_σ^s and P_σ^u, which is the same as the minimal

objective value of $\hat{P}_2$, provides a lower bound of T_σ.

For problem P_σ^s, since $p_{[i-2],3}$, i=2, ...,k and $p_{[i-1],2}$, $p_{[i],1}$, i=1, ..., k are all known for the given σ and $w_{-1,3}$ (i.e., $w_{n-1,3}$) only appears in constraint (8a) for i=2 and in constraint $w_{-1,3} \geq 0$, an optimal solution of P_σ^s can then be obtained in the following recursive way:

$w_{-1,3} = 0$,
$w_{i-1,2} = \max\{0, p_{[i-1],2} - \mu_2 - w_{i-2,3}\}$, i=1, ..., k,
$w_{i,1} = \max\{0, p_{[i],1} - \mu_1 - w_{i-1,2}\}$, i=1, ..., k,
$w_{i-1,3} = \max\{0, p_{[i-1],3} - \mu_3 - w_{i,1}\}$, i=1, ..., k.

Therefore, problem P_σ^s can be solved in $O(k)$ time.

For problem P_σ^u, we can prove that there is an optimal solution $\{w_{i-2,3}, w_{i-1,2}, w_{i,1}$, i=k+1, ..., n$\}$ such that $w_{i-2,3} = 0$, i=k+1, ..., n, because otherwise, we can construct another solution $\{w'_{i-2,3}, w'_{i-1,2}, w'_{i,1}$, i=k+1, ..., n$\}$ such that $w'_{i-2,3} = 0$, $w'_{i-1,2} = w_{i-1,2} + w_{i-2,3}$ and $w'_{i,1} = w_{i,1}$. This solution satisfies all constraints of P_σ^u and has an optimal objective value as well. Therefore, problem P_σ^u is equivalent to the following problem $\tilde{P}_\sigma^u$:

$$\tilde{P}_\sigma^u: \quad \min \ (n-k)C + \sum_{i=k+1}^{n} (w_{i-1,2}+w_{i,1}) \qquad (9a)$$

subject to

$w_{i-1,2} \geq p_{[i-1],2} - \mu_2$, i=k+1, ..., n, (9b)
$w_{i,1} + w_{i-1,2} \geq p_{[i],1} - \mu_1$, i=k+1, ..., n, (9c)
$w_{i,1}, w_{i-1,2} \geq 0$, i=k+1, ..., n. (9d)

The optimal solution of $\tilde{P}_\sigma^u$ must have:

$w_{i-1,2} = \max\{0, p_{[i-1],2} - \mu_2\}$, i=k+1, ..., n, (10a)
$w_{i,1} = \max\{0, p_{[i],1} - \mu_1 - w_{i-1,2}\}$
$= \max\{0, p_{[i],1} - \mu_1 - \max\{0, p_{[i-1],2} - \mu_2\}\}$, i=k+1, ..., n. (10b)

Thus, the optimal objective function value is :

$$(n-k)C + \sum_{i=k+1}^{n} (w_{i-1,2}+w_{i,1})$$

$$= (n-k)C + \sum_{i=k+1}^{n} (\max\{0, p_{[i-1],2} - \mu_2\} + \max\{0, p_{[i],1} - \mu_1 - \max\{0, p_{[i-1],2} - \mu_2\}\})$$

$$= (n-k)C + \sum_{i=k+1}^{n} \max\{\max\{0, p_{[i-1],2} - \mu_2\}, p_{[i],1} - \mu_1\}$$

$$= (n-k)(C-\mu_1-\mu_2) + \sum_{i=k+1}^{n} \max\{\max\{\mu_2, p_{[i-1],2}\} + \mu_1, p_{[i],1} + \mu_2\}$$

$$= (n-k)(C-\mu_1-\mu_2) + \sum_{i=k}^{n-1} \max\{\max\{\mu_2, p_{[i],2}\} + \mu_2, p_{[i+1],1} + \mu_2\}$$

$$= (n-k)(C-\mu_1-\mu_2) + \sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+1)})\}, \qquad (11)$$

where

$f_j = \max\{\mu_2, p_{j,2}\} + \mu_1$ and $g_j = p_{j,1} + \mu_2$, $j=1, 2, \ldots, n$.

Note that when k=0, the expression $\sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+1)})\}$ can be interpreted as the total length of tour σ of a traveling salesman problem (TSP) where the distance from city i to city j is defined by $d_{i,j} = \max\{f_i, g_j\}$. Therefore, problem $\tilde{P}_\sigma^u$ can be solved by using the algorithm of Gilmore and Gomory [2] in O(nlogn) time. When k>0, the expression $\sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+1)})\}$ can be interpreted as the total length of a path from city $\sigma(k)$ to city $\sigma(n)$ ($\sigma(k)$ and $\sigma(n)$ are both known) which passes exactly once through each city belonging to $\{1, \ldots, n\} \setminus \{\sigma(i), i \in S-\{n, k\}\}$. The shortest path can still be found using the Gilmore and Gomory algorithm after slight modification. For more details of this algorithm, please refer to Appendix 1.

(b) $LB_\sigma(2,3)$

In this case, M_2 and M_3 are taken as the two bottleneck machines. Note that T_σ can be rewritten as

$$T_\sigma = nC + \sum_{i=1}^{n} (w_{i,1} + w_{i-1,3} + w_{i,2}) \qquad (12)$$

and the constraints(5a) — (5g) can be rewritten as

$$w_{i,1} + w_{i-1,2} \geq p_{[i],1} - \mu_1, \qquad i=1, \ldots, n, \tag{13a}$$
$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \qquad i=1, \ldots, n, \tag{13b}$$
$$w_{i,2} + w_{i-1,3} \geq p_{[i],2} - \mu_2, \qquad i=1, \ldots, n, \tag{13c}$$
$$w_{i,1}, w_{i-1,3}, w_{i,2} \geq 0, \qquad i=1, \ldots, n. \tag{13d}$$

Using a similar approach as in (a), if we drop constraints (13a) for i=1 and i=k+1, ..., n, i.e., if we relax the processing time requirements of parts $\sigma(i)$, i=1, k+1, k+2, ..., n on machine 1, we can obtain a relaxed problem of P_2 (see Section 2 (a)) which can be decomposed into the following two subproblems P_σ^s and P_σ^u:

P_σ^s: $\min kC + \sum_{i=1}^{k} (w_{i,1}+w_{i-1,3}+w_{i,2})$

subject to (13a), i=2, ..., k; (13b)—(13d), i=1, ..., k.

and

P_σ^u: $\min (n-k)C + \sum_{i=k+1}^{n} (w_{i,1}+w_{i-1,3}+w_{i,2})$

subject to (13b) — (13d), i=k+1, ..., n.

The sum of the minimal objective values of P_σ^s and P_σ^u provides a lower bound for T_σ.

As in subsection (a), problem P_σ^s can be solved by using the following recursive formula in $O(k)$ time:

$$w_{1,1} = 0,$$
$$w_{i-1,3} = \max\{0, p_{[i-1],3} - \mu_3 - w_{i,1}\}, \qquad i=1, \ldots, k,$$
$$w_{i,2} = \max\{0, p_{[i],2} - \mu_2 - w_{i-1,3}\}, \qquad i=1, \ldots, k,$$
$$w_{i+1,1} = \max\{0, p_{(i+1),1} - \mu_1 - w_{i,2}\}, \qquad i=1, \ldots, k.$$

and problem P_σ^u is equivalent to the following problem $\tilde{P}_\sigma^u$:

$\tilde{P}_\sigma^u$: $\min n-k)C + \sum_{i=k+1}^{n} (w_{i-1,3}+w_{i,2})$ (14a)

subject to

$w_{i-1,3} \geq p_{[i-1],3} - \mu_3$, $\quad i=k+1, \ldots, n,$ (14b)

$w_{i,2} + w_{i-1,3} \geq p_{[i],2} - \mu_2$, $\quad i=k+1, \ldots, n,$ (14c)

$w_{i-1,3}, w_{i2} \geq 0$, $\quad i=k+1, \ldots, n.$ (14d)

Similarly to (a), the optimal objective function value is such that

$$(n-k)(C-\mu_2-\mu_3) + \sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+1)}\}, \tag{15}$$

where

$f_j = \max\{\mu_3, p_{j,3}\}+\mu_2$ and $g_j = p_{j,2}+\mu_3$, j=1, 2, ..., n.

Therefore, problem $\tilde{P}_\sigma^u$ can also be solved by using the Gilmore and Gomory algorithm.

(c) $LB_\sigma(1,3)$

In this case, M_1 and M_3 are taken as the two bottleneck machines. Note that T_σ can be rewritten as

$$T_\sigma = nC + \sum_{i=1}^{n} (w_{i-1,2}+w_{i,1}+w_{i-1,3}) \tag{16}$$

and the constraints (5a) — (5g) can be rewritten as

$w_{i-1,2} + w_{i-2,3} \geq p_{[i-1],2} - \mu_2$, $\quad i=1, \ldots, n,$ (17a)

$w_{i,1} + w_{i-1,2} \geq p_{[i],1} - \mu_1$, $\quad i=1, \ldots, n,$ (17b)

$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3$, $\quad i=1, \ldots, n,$ (17c)

$w_{i-1,2}, w_{i,1}, w_{i-1,3}, \geq 0$, $\quad i=1, \ldots, n.$ (17d)

Using a similar approach as in (a), if we drop constraints (17a) for i=1 and i=k+1, ..., n, i.e., if we relax the processing time requirements of parts σ(i), i=k, ..., n on machine 2, we can obtain a relaxed problem of P_2, which can be decomposed into the following two subproblems P_σ^s and P_σ^u:

P_σ^s: $\quad \min kC + \sum_{i=1}^{k} (w_{i-1,2}+w_{i,1}+w_{i-1,3})$

subject to (17a), i=2, ..., k; (17b)—(17d), i=1, ..., k,

and

$$P_\sigma^u: \quad \min (n-k)C + \sum_{i=k+1}^{n} (w_{i-1,2}+w_{i,1}+w_{i-1,3})$$

subject to (17b)— (17d), i=k+1, ..., n.

The sum of the minimal objective values of P_σ^s and P_σ^u provides a lower bound for T_σ.

As in subsection (a), problem P_σ^s can be solved by using the following recursive formula in $O(k)$ time:

$$w_{0,2} = 0,$$
$$w_{i,1} = \max\{0, p_{[i],1} - \mu_1 - w_{i-1,2}\}, \quad i=1, \ldots, k,$$
$$w_{i-1,3} = \max\{0, p_{[i-1],3} - \mu_3 - w_{i,1}\}, \quad i=1, \ldots, k,$$
$$w_{i,2} = \max\{0, p_{(i),2} - \mu_2 - w_{i-1,3}\}, \quad i=1, \ldots, k,$$

and problem P_σ^u is equivalent to the following problem $\tilde{P}_\sigma^u$:

$$\tilde{P}_\sigma^u: \quad \min \ (n-k)C + \sum_{i=k+1}^{n} (w_{i,1}+w_{i-1,3}) \tag{18a}$$

subject to

$$w_{i,1} \geq p_{[i],1} - \mu_1, \qquad i=k+1, \ldots, n, \tag{18b}$$
$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \qquad i=k+1, \ldots, n, \tag{18c}$$
$$w_{i,1}, w_{i-1,3} \geq 0, \qquad i=k+1, \ldots, n. \tag{18d}$$

Similarly to (a), the optimal objective function value of $\tilde{P}_\sigma^u$ is

$$(n-k)(C-\mu_1-\mu_3) + \sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+1)}\}, \tag{19}$$

where

$f_j = p_{j,3}+\mu_1$ and $g_j = \max\{\mu_1, p_{j,1}\}+\mu_3$, j=1, 2, ..., n.

Therefore, problem $\tilde{P}_\sigma^u$ can also be solved by using the Gilmore and Gomory algorithm.

3.2. 3-machine robotic cell with robot sequence S_6

(a) $LB_\sigma(1,2)$

As in the case of robot sequence S_2, the lower bound can be computed using the Gilmore and Gomory algorithm.

(b) $LB_\sigma(2,3)$

As in the case of robot sequence S_2, the lower bound can be computed using the Gilmore and Gomory algorithm.

(c) $LB_\sigma(1,3)$

In this case, M_1 and M_3 are taken as the two bottleneck machines. Note that the constraints (6a) — (6g) can be rewritten as

$$w_{i-1,2} + w_{i-2,3} \geq p_{[i-1],2} - \mu_2, \qquad i=1, \ldots, n, \tag{20a}$$
$$w_{i,1} + w_{i-1,2} + w_{i-2,3} \geq p_{[i],1} - \mu_1, \qquad i=1, \ldots, n, \tag{20b}$$
$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \qquad i=1, \ldots, n, \tag{20c}$$
$$w_{i-1,2}, w_{i,1}, w_{i-1,3} \geq 0, \qquad i=1, \ldots, n. \tag{20d}$$

Similarly to subsection 3.1(c), if we drop constraints (20a) for i=1 and i=k+1, ..., n, i.e., relax the processing time requirements of parts $\sigma(i)$, i=k, ..., n on machine 2, we can obtain a relaxed problem of P_6, which can be decomposed into the following two subproblems P_σ^s and P_σ^u:

P_σ^s: min $kC + \sum_{i=1}^{k} (w_{i-1,2}+w_{i,1}+w_{i-1,3})$

subject to (20a), i=2, ..., k; (20b)—(20d), i=1, ..., k,

and

P_σ^u: min $(n-k)C + \sum_{i=k+1}^{n} (w_{i-1,2}+w_{i,1}+w_{i-1,3})$

subject to (20b) — (20d), i=k+1, ..., n.

The sum of the minimal objective values of P_σ^s and P_σ^u provides a lower bound for T_σ.

As in subsection 3.1(c), problem P_σ^s can be solved using a recursive formula in $O(k)$ time. However, problem P_σ^u seems to be more complicated. With a similar argument as before, we can show that problem P_σ^u is equivalent to the following problem $\tilde{P}_\sigma^u$:

$$\tilde{P}_\sigma^u: \quad \min (n-k)C + \sum_{i=k+1}^{n} (w_{i,1}+w_{i-1,3}) \tag{21a}$$

subject to

$$w_{i,1} + w_{i-2,3} \geq p_{[i],1} - \mu_1, \qquad i=k+1, \ldots, n, \tag{21b}$$

$$w_{i-1,3} + w_{i,1} \geq p_{[i-1],3} - \mu_3, \qquad i=k+1, \ldots, n, \tag{21c}$$

$$w_{i,1}, w_{i-1,3} \geq 0, \qquad i=k, \ldots, n. \tag{21d}$$

The difference between this problem $\tilde{P}_\sigma^u$ here and the one in subsection 3.1(c) is that constraint (21b) and constraint (21c) are interdepent while constraint (18b) in subsection 3.1(c) is independent of constraint (18c).

Remark 3.1:

Problem $\tilde{P}_\sigma^u$(21a—21d) is likely to be NP-hard.

Remark 3.2:

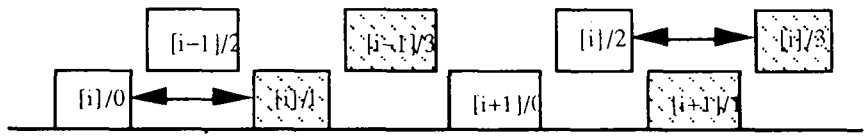
Since problem $\tilde{P}_\sigma^u$(21a-21d) is likely to be NP-hard, when we implement the lower bound scheme proposed at the beginning of this section for robot sequence S_6, we take $\Omega = \{(1,2), (2,3)\}$.

Remark 3.3:

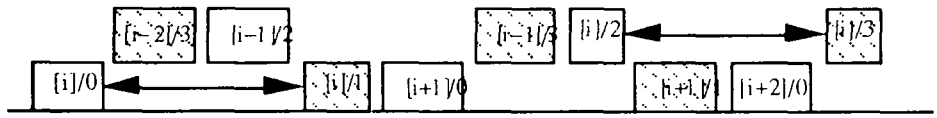
Problem $\tilde{P}_\sigma^u$(21a—21d) is more complicated to solve than other counterpart studied before, because although the processing time requirements of jobs on machine 2 is relaxed, i.e., the robot waiting time at machine 2 is zero, the robot waiting time at machine 1 and that at machine 3 are still interdependent. This situation is different from all others we have considered so far. This difference can be made much more clear by observing Figure 6.

Figures 6(a) and 6(b) are a portion of the time diagram in Fig. 4 and a portion of the time diagram in Fig. 5 respectively, where crosshatched boxes indicate possible robot waiting events before the robot performs corresponding moves. In Fig. 6(a), the robot waiting time at machine 1 is independent of the robot waiting time at machine 3 because there is no

crosshatched box between box [i]/0 and box [i]/1. However, in Fig. 6(b), the robot waiting waiting time at machine 1 is dependent on the robot waiting time at machine 3 and vice versa, because there is a crosshatched box [i-2]/3 (which is related to the robot waiting time at machine 3) between box [i]/0 and box [i]/1 and a crosshatched box [i+1]/1 (which is related to the robot waiting time at machine 1) between box [i]/2 and box [i]/3.



(a) Robot sequence is S_2



(b) Robot sequence is S_6

Fig. 6. Robot waiting events when M_1 and M_3 are taken as bottleneck machines

3.2. m-machine robotic cell (m>3) with an arbitrary robot sequence

In this subsection, we extend the lower bound scheme introduced at the beginning of this section to m-machine robotic cell (m>3) with an arbitrary robot sequence.

Suppose that the robot sequence we consider for an m-machine robotic cell is $\sigma_R = \alpha_{i_0} \alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_m}$, where $i_0=0$, i_1, i_2, ..., i_m is a permutation of 1, 2, ..., m, α_j, j=1, ..., m are as defined before. σ_R can be thought as a string of characters α_j, j=0, ..., m. We consider the robot sequence in two adjacent robot move cycles, i.e., $\sigma_R^2 = \alpha_{i_0} \alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_m} \alpha_{i_0} \alpha_{i_1} \alpha_{i_2} \ldots \alpha_{i_m}$.

A sub-sequence of σ_R^2 that starts and ends with α_{j-1} and α_j is denoted by $\sigma_R(j)$. The sub-sequence is unique if we neglect where it is located in σ_R^2. $\hat{\sigma}_R(j)$ is defined as the sub-sequence of $\sigma_R(j)$ such that $\sigma_R(j) = \alpha_{j-1}\hat{\sigma}_R(j)\alpha_j$, i.e., $\hat{\sigma}_R(j)$ is obtained from $\sigma_R(j)$ by eliminating the first character α_{j-1} and the last character α_j.

For i, j$\in$ {1, 2, ..., m}, i$\neq$j, we say that machine M_i and machine M_j are nested into each other if $\hat{\sigma}_R(i)$ includes α_j and $\hat{\sigma}_R(j)$ includes α_i. For instances, for robot sequence S_2 of 3-

machine robotic cells, we have $\sigma_R^2 = \alpha_0\alpha_2\alpha_1\alpha_3\alpha_0\alpha_2\alpha_1\alpha_3$; $\sigma_R(1) = \alpha_0\alpha_2\alpha_1$, $\hat{\sigma}_R(1) = \alpha_2$ and $\sigma_R(3) = \alpha_2\alpha_1\alpha_3$, $\hat{\sigma}_R(3) = \alpha_1$. Since $\hat{\sigma}_R(1)$ does not include α_3, machine M_1 and machine M_3 are not nested into each other. However, for robot sequence S_6, we have $\sigma_R^2 = \alpha_0\alpha_3\alpha_2\alpha_1\alpha_0\alpha_3\alpha_2\alpha_1$; $\sigma_R(1) = \alpha_0\alpha_3\alpha_2\alpha_1$, $\hat{\sigma}_R(1) = \alpha_3\alpha_2$ and $\sigma_R(3) = \alpha_2\alpha_1\alpha_0\alpha_3$, $\hat{\sigma}_R\,\hat{\sigma}_R(3) = \alpha_1\alpha_0$. Since both $\hat{\sigma}_R(1)$ includes α_3 and $\hat{\sigma}_R(3)$ includes α_1, machine M_1 and machine M_3 are nested into each other.

For an m-machine robotic cell with an arbitrary robot sequence σ_R, the following Proposition holds:

Proposition 3.1. For any $j \in \{1, 2, \ldots, m-1\}$, machine M_j and machine M_{j+1} are not nested into each other.

For any $r, s \in \{1,2,\ldots,m\}$, $r<s$, we consider the lower bound $L_\sigma(r, s)$ for partial schedule $\sigma = (\sigma(i), i \in S)$, $S = \{n, 1, 2, \ldots, k\}$. Then Proposition 3.2 holds:

Proposition 3.2. If machine M_r and machine M_s are not nested into each other, then the optimal objective function value of the corresponding problem $\tilde{P}_\sigma^u$ as defined for 3-machine robotic cells in subsections 3.1 and 3.2 is a constant (independent of the part sequence). It can be represented as $A + \sum_{i=k}^{n-1} \max\{f_{\sigma(i)}, g_{\sigma(i+h)}\}$, where A is a constant, h is a positive integer constant, and $f_j, g_j \geq 0$, $j=1, \ldots, n$ are defined by the parameters δ_i^-, δ_i^+, $i=0, 1\ldots, m+1$, θ_{ij}, $i, j=0, 1, \ldots, m+1$ and p_{ij}, $i=1, 2, \ldots, n$, $j=1, 2,\ldots, m$ of the robotic cell.

Proposition 3.3. For any $j \in \{1, 2, \ldots, m-1\}$, $L_\sigma(j, j+1)$ can be computed by using the Gilmore and Gomory algorithm.

For proofs of the above three propositions please refer to Appendix 2.

According to Proposition 3.2, for m-machine ($m>3$) robotic cells with arbitarily robot sequence, we can implement the lower bound scheme $LB_\sigma(\Omega) = \max\{LB_\sigma(u,v) \mid (u, v) \in \Omega\}$ where $\Omega = \{(1, 2), (2, 3), \ldots, (m-1, m)\}$. The lower bound $LB_\sigma(\Omega)$ can be computed by using the Gilmory and Gilmore algorithm in O(mnlogn) time.

4. Branch and Bound Algorithm

We now present the main features of our branch and bound algorithm.

The algorithm first arbitarily specifies the job scheduled at the last position (the n-th position) and then enumerates the jobs at positions 1, 2, ..., n-1 successively.

An initial upper bound is obtained by comparing the cycle times of the part sequences generated when we compute a lower bound for the root node for each choice of the two bottleneck machines by using the Gilmore and Gomory algorithm. The cycle times can be computed by using a graph-based algorithm proposed by Chen, Chu and Proth [1] or by solving a linear programming problem.

For nodes that are not eliminated in the enumeration tree of the algorithm, a lower bound is computed by using the method proposed in Section 3. When a complete schedule is reached, its cycle time is computed by using the graph-based algorithm or by solving a linear programming problem, which is compared with the best upper bound obtained up to now. If the cycle time is less than the upper bound, the upper bound is updated and a new upper bound is obtained. In the algorithm, we use the depth-first-search plus best lower bound rule to select which node should be examined next.

In addition, some eliminating rules can be implemented in our algorithm by taking into account the feature of the minimal part set.

5. Computational Experiments

We have tested the performance of our algorithm using randomly generated problems for 3-machine robotic cells with given robot sequences S_2 and S_6. The problem data are generated in the following way: $\delta_i^- = \delta_i^+ = 2$, i=0, 1, ..., m+1; $\theta_{ij} = 4|i-j|$, i, j = 0, 1, ..., m+1; p_{ij}, i=1, ..., n, j=1, ..., m, the processing times of parts on machines, are generated by using the scheme given by Lageweg, Lenstra, and Rinnooy Kan [7], i.e., four different problem classes are considered: random problems, problems with correlation between the processing times of each job, problems for which the processing times of each job have a positive trend, and problems with both correlation and positive trend. Table 1 presents the distributions of these classes, where c(i) is the correlation coefficient of job i, i =1, ..., n, taken from a uniform [0, 4] distribution, and j is the machine index. All parts in MPS are considered to be different, i.e., no MPS-based eliminating rules are used in our computational experiments.

Table 1. Problem classes

Problem class	Distribution	Parameters
Random (R)	Uniform	1, 100
Correlation (C)	Uniform	20c(i)+1, 20c(i)+20
Trend (T)	Uniform	12.5(j-1)+1, 12.5(j-1)+100
Correlation/trend (CT)	Uniform	2.5(j-1)+20c(i)+1, 2.5(j-1)+20c(i)+20

Table 2 presents the computational results of our algorithm for 3-machine robotic cells with robotic sequences S_2 and S_6 and with n=10 and n=15, where column "CPU seconds" denotes the average CPU time in seconds on a RISC 6000 for 10 randomly generated instances of each problem class. It should be noted that the average computation time for problem classes with robot sequence S_6 is much less than that for these with robot sequence S_2.

Table 2. Performance of the branch-and-bound algorithm for n=10, 15

$\sigma_R = S_2$			$\sigma_R = S_6$		
Parameters	Problem class	CPU seconds	Parameters	Problem class	CPU seconds
	R	29.0		R	1.83
n = 10	C	24.6	n = 10	C	5.61
	T	36.5		T	1.27
	CT	31.9		CT	3.82
	R	*		R	150.5*
n = 15	C	*	n = 15	C	*
	T	241.1		T	87.5
	CT	*		CT	*

* problem classes for which the average computation time exceeds 300 seconds.

In order to illustrate relative sharpness of the lower bound, we have run our optimal algorithm as a heuristic for large size problems (n=20 and n=30). Table 3 presents the performance of the heuristic run of our optimal algorithm. Column 2 in the table describes the stopping criterion used in the heuristic run. For example, in 20-part problems, we terminate the optimal algorithm when the total CPU time exceeds 60 seconds or when we find an upper bound which is within 5% of the lower bound. It is interesting to point out that the average relative difference of the upper bound and the lower bound obtained when the algorithm is terminated is less than 5% for problem classes with robot sequence S_6 and less than 10% for these with robot sequence S_2.

Table 3. Heuristic run of the branch and bound algorithm

Parameters	Stopping criterion	Problem class	CPU seconds	(UB-LB)/LB
		R	40.03	0.0705
$\sigma_R = S_2$	CPU ≥ 45s or	C	23.47	0.0543
n = 15	(UB-LB)/LB≤0.05	T	27.56	0.0550
		CT	36.258	0.0612
		R	0.560	0.0332
$\sigma_R = S_6$	CPU ≥ 45s or	C	0.445	0.0399
n = 15	(UB-LB)/LB≤0.05	T	0.304	0.0219
		CT	0.416	0.0358
		R	60.00	0.0612
$\sigma_R = S_2$	CPU ≥ 60s or	C	60.00	0.0733
n = 20	(UB-LB)/LB≤0.05	T	48.23	0.0654
		CT	60.00	0.0684
		R	1.699	0.0327
$\sigma_R = S_6$	CPU ≥ 60s or	C	0.996	0.0266
n = 20	(UB-LB)/LB≤0.05	T	0.925	0.0253
		CT	0.831	0.0226
		R	90.00	0.0886
$\sigma_R = S_2$	CPU ≥ 90s or	C	73.93	0.0703
n = 30	(UB-LB)/LB≤0.05	T	69.43	0.0645
		CT	74.02	0.0670
		R	4.720	0.0378
$\sigma_R = S_6$	CPU ≥ 90s or	C	2.640	0.0256
n = 30	(UB-LB)/LB≤0.05	T	1.695	0.0275
		CT	2.462	0.0264

6. Conclusion

In this paper we propose a general bounding scheme for the part sequencing problem in robotic cells processing multiple types of parts when the robot sequence is given. In the bounding scheme, a lower bound can be quickly computed by using the well-known Gimory and Gilmore algorithm. We propose a branch-and-bound algorithm based on the bounding scheme. The performance of the algorithm is given for 3-machine robot cells with given robot sequences S_2 and S_6 and with n=10, 15. The heursitic run of the branch-and-bound algorithm

shows that the average relative difference of the upper bound and the lower bound obtained when the algorithm is terminated is less than 5% for problem classes with robot sequence S_6 and less than 10% for these with robot sequence S_2.

Further investigation is needed to develop much sharper lower bounds which can be computed in polynomial time and to develop eliminating rules based on dominance properties to attack large-size problems.

References

[1] Chen Haoxun, Chu Chengbin, J.M. Proth, Cyclic Scheduling of a hoist with time window constraints, submitted for publication in IEEE Transactions on Robotics and Automation, also Rapport de Recherche, No. 2307, 1994, INRIA-Lorraine, France.

[2] P.C. Gilmore and R.E. Gomory, Sequencing a one state-variable machine: A solvable case of the traveling salesman problem, *Operations Research*, Vol. 12, pp. 655-679, 1964.

[3] N. G. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: Two machine cells and identical parts, Working paper #93-06, Dept. of Industrial Engineering, University of Toronto, Canada, 1993.

[4] N. G. Hall, H. Kamoun, C. Sriskandarajah, Scheduling in robotic cells: Large cells, Working paper #93-07, Dept. of Industrial Engineering, University of Toronto, Canada, 1993.

[5] H. Kamoun, N. G. Hall, C. Sriskandarajah, Scheduling in robotic cells: Heuristics and cell design, Working paper #93-08, Dept. of Industrial Engineering, University of Toronto, Canada, 1994.

[6] K. Kamoun and C. Sriskandarajah, The complexity of scheduling jobs in repetitive manufacturing systems, *European Journal of Operational Research* 70(1993) 350-364.

[7] B. J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan, A general bounding scheme for the permutation flow-shop problem, *Operations Research*, Vol. 26, No. 1, pp. 53-67, 1978.

[8] S.P. Sethi, et al., Sequencing of parts and robot moves in a robotic cell, *Int. J. of Flexible Manufacturing Systems*, 4(1992): 331-358.

Appendix 1

Finding a shortest path by using the Gilmore and Gomory algorithm:

Problem P:

Given n cities 1, ..., n where the distance from city i to city j is defined by $d_{ij} = \max\{f_i, g_j\}$, f_i, $g_j > 0$, find a shortest path from city 1 to city n which passes each city exactly once.

We introduce another related problem P':

Problem P':

Given n+1 cities 1, ..., n, n+1 where the distance from city i to city j is defined by $d_{ij} = \max\{f_i, g_j\}$, where f_i, i=1,..., n-1, g_j, j=1, ..., n are the same as those in Problem P, $g_1 = f_n = g_{n+1} = f_{n+1} = L>0$, find a shortest tour of the n+1 cities.

Property A1:

Let $L_{max}(P)$ denote the length of the longest paths that pass each city once for Problem P. If $L \geq L_{max}(P)$, then any shortest tour of Problem P' contains both the arc from city n to n+1 and the arc from city n+1 to city 1.

Proof: For any tour containing the two arcs, the length is no larger than $L_{max}(P) + 2L$, but for any tour which does not contain at least one of the arcs, the length is larger than 3L. Since $L_{max}(P) + 2L \leq 3L$, the assertion of Property A1 is rue. Q.E.D.

Property A2:

Suppose that the condition of Property A1 is satisfied and that $1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{n-1} \rightarrow n \rightarrow n+1 \rightarrow 1$ is a shortest tour of Problem P' (where i_2, ..., i_{n-1} is a permutation of 2, ..., n-1), then $1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{n-1} \rightarrow n$ is a shortest path of Problem P.

Proof: If $1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{n-1} \rightarrow n$ is not a shortest path of P, then there is a path $1 \rightarrow j_2 \rightarrow \ldots \rightarrow j_{n-1} \rightarrow n$ (j_2, ..., j_{n-1} is a permutation of 2, ..., n-1) such that $L_2 < L_1$, where L_1 (resp. L_2) denotes the length of the path $1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{n-1} \rightarrow n$ (resp. path $1 \rightarrow j_2 \rightarrow \ldots \rightarrow j_{n-1} \rightarrow n$).

Let us denote by C_1 (resp. C_2) the length of the tour $1 \rightarrow i_2 \rightarrow \ldots \rightarrow i_{n-1} \rightarrow n \rightarrow n+1 \rightarrow 1$ (resp. tour $1 \rightarrow j_2 \rightarrow \ldots \rightarrow j_{n-1} \rightarrow n \rightarrow n+1 \rightarrow 1$). Obviously, $C_1 = L_1 + d_{n,n+1} + d_{n+1,1}$, $C_2 = L_2 + d_{n,n+1} + d_{n+1,1}$. It follows that $C_2 < C_1$, which is contrary to the assumption of the property. Q.E.D.

The above two properties imply that we can solve Problem P by solving Problem P'. Since

Problem P' can be solved by using the Gilmore and Gomory algorithm, so can be Problem P.

Appendix 2

Proof of Proposition 3.1:

Two cases may happen for α_j and α_{j+1} in σ_R. One is that α_j occurs before α_{j+1} in σ_R. The other is that α_{j+1} occurs before α_j.

For the former case, the sequences σ_R^2 can be represented by $\sigma_1\alpha_j\sigma_2\alpha_{j+1}\sigma_3\sigma_1\alpha_j\sigma_2\alpha_{j+1}\sigma_3$, where σ_1, σ_2 and σ_3 are sub-sequences of σ_R, which do not include α_j and α_{j+1}. Clearly, $\hat{\sigma}_R(j+1) = \sigma_2$. Since σ_2 does not include α_j, neither does $\hat{\sigma}_R(j+1)$, so machine j and machine j+1 are not nested into each other.

For the latter case, the sequences σ_R^2 can be represented by $\sigma_1\alpha_{j+1}\sigma_2\alpha_j\sigma_3\sigma_1\alpha_{j+1}\sigma_2\alpha_j\sigma_3$, where σ_1, σ_2 and σ_3 are sub-sequences of σ_R, which do not include α_j and α_{j+1}. Clearly, $\hat{\sigma}_R(j+1) = \sigma_3\sigma_1$. Since $\sigma_3\sigma_1$ does not include α_j, neither does $\hat{\sigma}_R(j+1)$, so machine j and machine j+1 are not nested into each other.

Proof of Proposition 3.2:

Since machine M_r and machine M_s are not nested into each other, either $\hat{\sigma}_R(r)$ does not include α_s or $\hat{\sigma}_R(s)$ does not include α_r.

Case 1: neither $\hat{\sigma}_R(r)$ includes α_s nor $\hat{\sigma}_R(s)$ includes α_r

In this case, since the robot waiting times at all machines except M_r and M_s are zero and $\hat{\sigma}_R(r)$ does not include α_s, there is no robot waiting event occurred between the time that a part is loaded onto machine M_r and the time that the part finishes its processing on M_r. Therefore, the robot waiting time at machine M_r related to part [i] can be represented by $\max\{0, p_{[i],r} - \mu_r\}$, where μ_r is a constant, which is the busy time of the robot between these two times. Similarly, the robot waiting time at machine M_s related to part [i] can be represented by $\max\{0, p_{[i],s} - \mu_s\}$. Notice that the sum of $\max\{0, p_{[i],r} - \mu_r\}$ (resp. $\max\{0, p_{[i],s} - \mu_s\}$) for all unscheduled parts is a constant (independent of the part sequence), the optimal objective of $\tilde{P}_\sigma^u$ is a constant (independent of the part sequence).

Case 2: $\hat{\sigma}_R(r)$ does not include α_s but $\hat{\sigma}_R(s)$ includes α_r

In this case, similarly to Case 1, the waiting time at machine M_r related to job [i] can be represented by $w_{i,r} = \max\{0, p_{[i],r} - \mu_r\}$. Let [j]/s denote the robot move of type [.]/s immediately following the robot move [i]/r. Since $\hat{\sigma}_R(s)$ includes α_r, [i]/r is located between the moves [j]/s-1 and [j]/s in the time diagram of the robotic cell and j=i+h, where h = h'+1, and h' is the number of parts on machines r+1, ..., s-2 when the robot performs the move [j]/s-1. Due to one-unit cyclic nature of the schedules under consideration, h' is constant

(independent of i) and so is h. Moreover, since the robot waiting times at all machines except M_r and M_s are zero, [i]/r is the unique move between the moves [j]/s-1 and [j]/s such that the robot may wait before performing it. Therefore, the robot waiting time related to [j]/s ([i+h]/s) can be represented by $w_{i+h,s} = \max\{0, p_{[i+h],s} - \mu_s - w_{i,r}\}$ and the result of the proposition can be derived as discussed in subsections 3.1 and 3.2 for 3-machine robotic cells.

Case 3: $\hat{\sigma}_R(s)$ does not include α_r but $\hat{\sigma}_R(r)$ includes α_s.

Similar to Case 2.

Proof of Proposition 3.3:

This proposition directly follows from Prositions 3.1 and 3.2 and the fact h' = 0 for r=j, s=j+1 in the proof of Proposition 3.2.

Unité de recherche INRIA Lorraine
Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)

Unité de recherche INRIA Rennes - IRISA, Campus universitaire de Beaulieu 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes - 46, avenue Félix Viallet - 38031 Grenoble Cedex 1 (France)
Unité de recherche INRIA Rocquencourt - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis - 2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249 - 6399